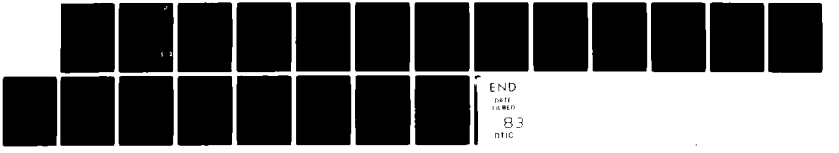
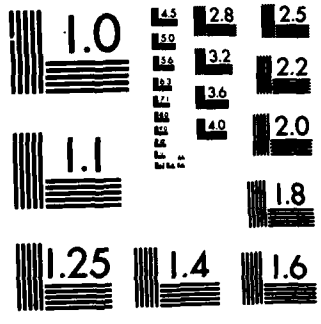
AD-A123 910 TIME AND PRODUCTION SYSTEMS(U) DREXEL UNIV PHILADELPHIA 1/1
PA R M SALTER 02 DEC 81 DREXEL-TR-2-81
N000014-80-C-0752

UNCLASSIFIED F/G 6/4 NL



END
DATE FILMED
83
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Technical Report #2-81

ONR Contract N00014-80-C-0752

Time and Productions Systems

Richard M. Salter

Drexel University

1981

TIME AND PRODUCTION SYSTEMS

Richard M. Salter

Drexel University

ABSTRACT

The utility of rule based production systems seems to be in the ability to model domains consisting of independent states and actions, in which process interaction is minimal. This methodology is suitable for domains outside of the realm of human information processing, in particular, in representing domains of continuous processes. We apply it to the design of a world modeling system, where objects and relationships are observed over time. We consider the effects of embedding time into a rule based system, where individual productions represent instantaneous events which are grouped together to form processes. We illustrate by discussing a particular implementation, the language CONCUR.

1. Introduction

The production system methodology has been successfully used in the design of systems to solve various problems in Artificial Intelligence, including expert systems, planning and problem solving, and theorem proving [Ni80]. It has been suggested that production systems are able to make more general use of the knowledge embedded within them because they lack a prior procedural orientation towards this knowledge. As a result, such systems are especially useful in modeling domains in which a strong procedural correlation of data would not naturally appear [Da76]. The production system is therefore an appropriate tool in the design of models for systems of independent processes whose interactions can be defined in terms system state. An example of this sort of application can be found in [Zis76], in which a modified production system is used to model the independent asynchronous concurrent processes which are part of an office environment.

In general, we would like to consider the impact of the production system methodology on the design of "world models", that is, sets of objects and relationships that vary over time. Such a model is used, for example, in discrete robot planning systems such as STRIPS [Fi72]. We are particularly interested in the use of production systems in the implementation of the symbolic discrete event

simulation [Zie76] system described in [He73] and [Lo77]. A principal feature of this simulation technique is its ability to model continuous and concurrent processes using operators acting on data structure representing the state of the world. A rule based model seems appropriate since the evolution of the domain in time depends both on continuously changing parameters, and on the action of discrete asynchronous events. The precondition-state transformation structure for rules permits the modeling of discrete events as a conditional set of actions. Processes, with which we associate duration, can be composed of sequences of events occurring at well defined times. A rule based system seems to be the natural representation of a set of loosely coupled processes whose interactions come about indirectly as a result of their effects on the system state.

Certain other advantages of rule based systems seem to reinforce its applicability to this problem domain. The large degree or modularity of simulation programs written using a rule based system would allow the user easy access to the various complexities of the modeled world. The control structure permits the actions of different processes to interleave arbitrarily without prior control specification by the user. Processes can be maintained as unified activities without resorting to procedural representations, and so permitting a larger range of possible interactions.

In this paper we will discuss the design of a rule based simulation system. The principal feature from the point of view of production system methodology is the introduction of a continuous term, in this case time, as a state parameter. We will discuss how knowledge about time can be embedded into a rule based system; how such knowledge affects the structure of the database, the rule set, and the system control structure. The techniques described in this paper have been used to implement the CONCUR world modeling system [Sal80], which is based upon the ideas first presented in [He73]. We will illustrate using CONCUR and a sample world, containing a robot, a bucket, and a water tap, which has been discussed in detail in [He73], [Lo77], and [Sal80] (see Figure 1).

2. Domain Representation

The skeletal structure of the CONCUR system is close to the classical production system formulation. Scenarios (rules) contain initiation conditions (antecedents) which are logical templates representing a set of possible states of the world model database. An event occurs when the initiation conditions of some scenario are successfully matched against the world, and the instantaneous effects (consequents) of an event can alter the database and the scenario set.

As in the planning system STRIPS [Fi72], the representation of the world model consists of a set of relations which are declarative and state-dependent. Unlike STRIPS, many attributes describe the world using numerical parameters. Thus (AT ROBBY 10), (AT TAP 15), (GOTO ROBBY 15), (GRASPING ROBBY PAIL), (CONTENTS PAIL 10) can be used to define the state of the world in terms of attribute value parameters. The database can always be used to obtain a shapshot of the world at some point in model time.

In any model there will generally be attributes whose values are not constant in time. We can represent those attributes by extending the syntactic range of the attribute values to include expressions. Thus, in CONCUR syntax, (AT ROBBY (* (PLUS (TIMES 3 (SUB TIME 3.1)) 10))) is also a legal entry, which can be interpreted as a trajectory for the location of a robot ROBBY over some range of model times (the free variable TIME refers to the model clock). Evaluating these expressions using the current model time produces the snapshot value of the world. This syntax permits the representation of an infinite number of individual states of infinitesimal duration as extensional knowledge within the database.

The active components of the system are a set of STRIPS-type rules representing instantaneous changes to the world. These "events" are organized into sequential processes by scenarios which are, in effect, "super rules"

whose preconditions coincide with the preconditions of the first event in the process, and whose effects are mapped out over time. For example, the GOTO scenario consists of two events which correspond to the beginning and end of the process of moving from one location to another. The first event starts the motion by inserting an expression into the location attribute, and the second stops it by replacing the expression with a constant. The actual motion can be conceptualized as a state of the system, as represented by the expression itself. This paradigm allows us to define a process occuring over some time duration in terms of the instantaneous changes to the trajectories of the parameters affected by process. The world model itself is then defined by a set of parameters whose trajectories are modified at discrete points in time by sequences of instantaneous actions.

In CONCUR, the TIME parameter is ultimately the only variable appearing in database expressions. This is because scenarios creating expressions unwind all other variables instantiated in the matching process until their (possibly time dependent) values are arrived at. The template for creating the above item could be (AT *ROBOT (|* (PLUS (TIMES *SPEED (SUB TIME *TIME)) *LOC))), where ROBOT, SPEED and LOC are bound by initiation conditions, and *TIME refers to the current point in model time (i.e. the time at which the expression is being created). We create the new database item by "quasi-quoting" (also known as "backquote", see

[Ch80]) this expression. This means that only terms preceeded by an asterisk are evaluated. The evaluation of such terms is complete in the current model, i.e. if any term encountered in the evaluation process evaluates to a formula, then that formula is also evaluated. The "|*" preceeding the new expression serves to "quote" the asterisk, so that it will be passed into the new assertion representing an evaluation call for the entire expression. Templates such as these permit new trajectories to be contingent on the values of attributes at the point in time at which the event creating the trajectories occurs.

The use of discrete events to redefine trajectories gives rise to time functions which are composed of piecewise smooth curves. It is possible to approximate the behavior of complex time functions with piecewise definitions using only a limited set of expressions by correcting the trajectories after small time intervals. We can, for example, approximate the sinusoidal time function of the simple harmonic motion of a spring using only polynomial expressions by feeding back position and accelleration data through discrete events occuring at fixed time intervals (see Figure 2). The complexity of the resulting approximations depends upon the repetoire of expressions permitted in the database. The set of allowable expressions is, however, limited by the control structure, as we shall see below.

3. Control Structure

In CONCUR, sets of sequentially ordered events are represented in a single scenario. The initiation conditions of the scenario are the preconditions of the first event of the process. When a scenario is invoked, this first event is fired, and a new scenario is created (if necessary) representing the rest of the process. The details of this representation are given below. Scenarios act as scripts for processes which may involve numerous events over some period of time. In most cases, representing processes using a single syntactic unit simplifies the model by permitting the logical grouping of sets of events that contribute to a single activity.

As we are dealing with a simulation system we shall only consider a control structure running in the forward mode. In the dual problem of planning, which is also under consideration, a more general control structure is required (more on this below). The basic cycle classically selects a single rule to invoke and re-evaluates the state of the system on each iteration. Since this state is parameterized by time, it becomes a required part of the selection process to determine when, in terms of the model clock, an event is to occur, and to order the selection in terms of these times.

This knowledge is embedded in the initiation conditions of a scenario both directly and indirectly. Initiation conditions are boolean expressions whose terms are either templates to be matched against the database, or predicate expressions which are to be evaluated in the environment produced by the match. The variable TIME is global to all such environments, and refers to current model time. It is thus possible to include a term such as (EQ TIME 10.2), from which it is easy to infer the invocation time of the event, but any scenario containing such a precondition is clearly of limited use. Other, more indirect terms such as (EQ LOC1 LOC2) or (LT LOC1 LOC2) will set bounds on invocation times. When all arguments to such predicates are constants, it is possible to infer that either they are satisfied in the current state, or some prior event must occur before they ever will become true. In the search for the next event, this analysis permits certain events to be ruled out. The more interesting case is when one or more of the arguments is represented by an expression, as we can now predict an exact time or interval of viability by solving similtaneous equations and/or inequalities. The pattern matcher is extended to include these more complex criteria.

In CONCUR, each instantiation of the initiation condition variables represents a potential event. During the matching process all predicate terms with variables bound to formulas remain unevaluated and are stored as part of the match environment. The result of a successful match

will include a set of bindings and possibly a set of equations and inequalities whose simultaneous solution will yield a time at which all conditions will be satisfied. An equation solver is used to determine a time interval (assuming consistency) during which the event may take place. Each potential event is therefore stamped with its earliest initiaiton time. The limitation on the mathematical content of the time expressions is imposed only by the capability of the equation solver. The problem of determining the set of equations to solve is entirely syntactic.

The initial part of the control cycle consists of associating binding sets with initiation times to determine the next event. It is possible that more than one event will be stamped with the same earliest time. Any such group of events can be regarded, from the point of view of the model, as potentially simultaneous. The way in which this is handled is a very important property of the model, and a thorough discussion is beyond the scope of this paper. It is worthwhile to note that to some extent this is a function of the application; for example one may want to follow all possible evolutions. It is not unrealistic to consider even a simple nondeterministic selection if the properties resulting from the choice are appropriate for the domain being modeled. It is useful, whenever possible, to design events which commute, so that conflict resolution will impact only those instances which are crucial to the

underlying model.

In the current CONCUR implementation, the scenario set has a static total ordering. Among potential events with the same earliest initiation time, the one associated with the least scenario in the total order is chosen. If more than one next event is associated with the same scenario, then the choice is nondeterministic (actually it is determined by the order in which the events were encountered by the pattern matcher, but there is no way that this can be controlled). The advantage of this is that by searching the scenario space in order, we can conclude the search immediately if any scenario binds an event which can initiate at the current model time. In any scheme for conflict resolution, the first criterion will always be initiation time.

When an event is chosen, the model clock is set to its initiation time (possibly the current time) and the body of its scenario is processed. The changes made by the invoked rule are specified using add and delete functions. If the process defined by the scenario contains additional events, then a new scenario is inserted representing the remainder of the process. A single control statement, CONTINU (called RESUME in [Sal80]), is used to separate the events of a given process, and appears in the scenario following the database changes. The format for a CONTINU argument is the scenario defining the rest of the process. The process will

not alter the database again until the preconditions of this scenario are satisfied. This scenario is placed in the scenario set and remains there until it is activated once ([Wat74] discusses the idea of rules creating new rules). In order for the new scenario to be able to refer to the bindings of this activation, we close the new scenario in that environment (see [St74] for a discussion of the use of closures to implement class instances). Closed scenarios replace the control blocks used in [He73] and [Lo77] as the activation record of a given process. Note that the contents of the executable part of a CONTINU argument may contain another CONTINU, allowing a single scenario to specify a sequence of events mapped out over time. (Clearly some sort of recursive specification is actually required here to express a possibly unbounded set of recurring events. This would involve a small extension to the implementation, but we have not yet settled on the appropriate syntax for expressing such a capability).

4. Conclusion

We have described a methodology for the design of world models which represent time dependent attributes as arithmetic expressions, and have used the production system formalism to manipulate this structure to produce discrete event simulations of continuous processes. Our representation presents processes as sequences of

instantaneous state changes, with the intervening continuous aspects represented directly in the database. This concept yields a uniform set of active functions (events), described simply by add-delete lists. which can create closures that serve as control blocks. Although individual events are tied together at a higher level, from the point of view of the database actions are discrete and correspond to the firing of production rules. It is in fact notable that the basic cycle of event driven simulation [Fr77, p.18] is analogous to the production system control structure once the mechanism for updating model time is introduced.

Our work in simulation is currently dealing with processes whose events need not necessarily be linearly ordered. This situation was dealt with in [Zis77] using Petri nets to describe the relationships between events. In such a system scenarios are used only to describe individual instantaneous events, while more complex procedural structures are used to organize these events into processes. Although ultimately events are fired in some total order, the user is freed from necessarily specifying that order a priori. We believe that this will give the user a larger degree of expressiveness in designing complex processes.

With world models we are dealing with numerous independent entities tied together by complex relationships and operated upon by independent processes. While the current application has been in the area of simulation,

there is no reason why the same basic structure cannot be used for other aspects of modeling, in particular to implement planning systems that are cognizent of model time. Such a system would require sets of rules which are dual to the events described above. The biggest problem results from the fact that simple equation solving can no longer be used to discriminate between rules, as such equations tend to contain too many uninstantiated terms. A higher level of reasoning is required for some sort of equation analysis. Aspects of this analysis will resemble the more classical techniques of discrete planning, such as regression [Wal77] and nonlinear and hierarchical planning [Sac77]. This research is currently in progress, and is reported on in [Sal81].

The methodology presented in this paper is an attempt to utilize the power of rule based systems in a design for symbolic world modeling, and a characterization of the requirements for applying such systems to problem domains which contain a continuous parameter, such as time. Our success with applying preliminary implementations of CONCUR to various different examples leads us to believe that the technique of embedding time used here is appropriate for expressing the dynamic evolution of these models.

REFERENCES

[Ch80] E. Charniak, C. Riesbeck, D. McDermott, "Artificial Intelligence Programming", L. Erlbaum Assoc., 1980.

[Da76] R. Davis And J. King, "An overview of production systems", Machine Intelligence, vol. 8 (1977), pp. 300-322.

[Fr77] W. R. Franta, "The Process View of Simulation ", Am. Elsivier, 1977.

[He73] G. Hendrix, "Modeling simultaneous actions and continuous processes", Artificial Intelligence, vol. 4 (1973), pp. 145-180.

[Lo77] J. Lowrance and D. P. Friedman, "Hendrix's model for simultaneous actions and continuous processes: an introduction and implementation", Int. J. Man-Machine Studies, vol. 9 (1977), pp 537-581.

[Sac77] E. D. Sacerdoti, "A Structure for Plans and Behavior", Am. Elsivier, 1977.

[Sal80] R. Salter, T. Brennan, and D. P. Friedman, "CONCUR: A language for continuous, concurrent processes", Computer Languages, vol. 5 (1980), pp. 163-189.

[Sal81] R. Salter, "Continuous planning -- an introduction", in preparation.

[St76] G. L. Steele, Jr., "LAMBDA: the ultimate declarative", MIT AI Memo 379, 1976.

[Wal77] R. Waldinger, "Achieving several goals simultaneously", Machine Intelligence, vol. 8 (1977), pp. 94-136.

[Wat74] D. A. Waterman, "Adaptive production systems", Complex Information Processing Working Paper No. 285, Psychology Dept., Carnegie-Mellon University, 1974.

[Zie76] B. P. Ziegler, "Theory of Modeling and Simulation", Wiley, 1976.

[Zis77] M. Zisman, "Use of production systems for modeling asynchronous, concurrent processes", in Pattern Directed Inference Systems, D. A. Waterman and F. Hayes-Roth, ed., Academic Press, (1977), pp. 53-68.

INITIAL DATABASE

```
(TYPE ROBBY ROBOT)                          (TYPE PAIL BUCKET)
(AT ROBBY 5)                                (GOTO ROBBY 10 2)
(GRASP ROBBY PAIL)                          (AT PAIL 8)
(FILL ROBBY PAIL VALVE 10 50)               (AT VALVE 10)
(CAPACITY PAIL 200)                         (CONTENTS PAIL 25)
(RELEASE ROBBY PAIL 10)
```

EVENT SCENARIOS

```
SCENARIO GOTO

Init. Conds.:   (GOTO =ROBOT =B =SPEED) (TYPE *ROBOT ROBOT) (AT *ROBOT =A)
Delete:         (GOTO *ROBOT *B *SPEED)
Subst:  (AT *ROBOT (!* (PLUS (MUL *SPEED (DIF TIME *TIME)) *A))
Continue     When:      (AT *ROBOT *B)
             Subst:     (AT *ROBOT ??)


SCENARIO TURN

Init. Conds.:   (TURN =ROBOT =V =R) (TYPE *ROBOT ROBOT) (AT *ROBOT =LOC)
                 (AT *V *LOC)
Delete:         (TURN *ROBOT *V *R)
Add:            (RATE *V *R)


SCENARIO FILL

Init. Conds.:   (FILL =ROBOT =BUCKET =V =LOC =R) (TYPE *ROBOT ROBOT)
                 (TYPE *BUCKET BUCKET) (AT *ROBOT *LOC) (AT *BUCKET *LOC)
                 (AT *V *LOC) (CAPACITY *BUCKET =CAP) (CONTENTS *BUCKET =CON))
Delete:         (FILL *ROBOT *BUCKET *V *LOC *R)
Add:            (TURN *ROBOT *V *R)
Continue     When:      (RATE *V *R)
             Subst:     (CONTENTS *BUCKET
                         (!* (PLUS (* (CONTENTS *BUCKET _))
                                   (MUL *R (DIF TIME *TIME))))
             Continue   When:    (!* (EQ (CONTENTS *BUCKET _) CAP))
                        Delete:  (RATE *V *R)
                        Add:     (RATE *V 0)
                        Subst:   (CONTENTS *BUCKET ??)
```

Figure 1 - CONCUR Robot-Bucket World

```
SCENARIO GRASP

Init. Conds.:   (GRASP =ROBOT =OBJ) (AT *ROBOT =LOC) (AT *OBJ *LOC)
Delete:         (GRASP *ROBOT *OBJ)
Add:            (GRASPING *ROBOT *OBJ)
Subst:          (AT *OBJ (^* (AT *ROBOT _)))


SCENARIO GRASPWATCH

Init. Conds.:   (GRASPING =ROBOT =OBJ) (AT *ROBOT =LOC1) (AT *OBJ =LOC2)
                 (\ (* (EQ^ LOC1 LOC2)))
Delete:         (AT *OBJ ??)
Add:            (AT *OBJ (^* (AT *ROBOT _)))


SCENARIO RELEASE

Init. Conds.:   (RELEASE =ROBOT =OBJ =X) (GRASPING *ROBOT *OBJ) (AT *ROBOT *X))
Delete:         (GRASPING *ROBOT *OBJ)
                 (RELEASE *ROBOT *OBJ *X))
Subst:          (AT *OBJ ??)
```

Fisure 1 - CONCUR Robot-Bucket World

INITIAL DATABASE

```
(AT SPRING1 1)                                  (ACCEL SPRING1 0)
(VELO SPRING1 0)                                (ADJ-ACC SPRING1)
```

EVENT SCENARIOS

SCENARIO SPRING-MOVE

```
Init. Conds.:   (AT =SPRING =LOC) (ACCEL *SPRING =A) (VELO *SPRING =V)
                 (ADJ-LOC *SPRING)
Delete:         (ADJ-LOC *SPRING)
                 (AT *SPRING *LOC)
Add:            (AT *SPRING
                    (!* (PLUS *LOC
                              (PLUS (MUL *V (SUB TIME *TIME))
                                    (DIV (MUL *A
                                              (MUL (SUB TIME *TIME)
                                                   (SUB TIME *TIME)))
                                         2)))))
Continue    When:       (!* (EQ TIME (PLUS *TIME 0.1)))
            Add:        (ADJ-ACC *SPRING)
```

SCENARIO SPRING-ACCEL

```
Init. Conds.:   (AT =SPRING =LOC) (ACCEL *SPRING =A) (VELO *SPRING =V)
                 (ADJ-ACC *SPRING)
Delete:         (ADJ-ACC *SPRING))
                 (ACCEL *SPRING *A)
                 (VELO *SPRING *V)
Add:            (ACCEL *SPRING (* (MINUS LOC))))
                 (VELO *SPRING (!* (PLUS *V (MUL (* (MINUS LOC))
                                                 (SUB TIME *TIME)))))
                 (ADJ-LOC *SPRING)
```

Figure 2 - Spring World

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Drexel Technical Report 2 - 81	2. GOVT ACCESSION NO. AD-A123 940	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Time and Production Systems		5. TYPE OF REPORT & PERIOD COVERED Technical Report #2 Dec 1981
		6. PERFORMING ORG. REPORT NUMBER ONR NR 049 - 480
7. AUTHOR(s) Richard M. Salter		8. CONTRACT OR GRANT NUMBER(s) N00014 - 80 - C - 0752
9. PERFORMING ORGANIZATION NAME AND ADDRESS Drexel University Philadelphia, PA 19104		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE 2 12/81
		13. NUMBER OF PAGES 17
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) ONR Code 411 IS Arlington, VA		15. SECURITY CLASS. (of this report) U
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Unlimited

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Rule-based systems, continuous processes

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper is concerned with the utility of rule-based production systems to model domains of continuous processes. In a world-modeling system, time is effected by individual productions grouped into processes representing instantaneous events. The resultant system is discussed in one particular implementation, the language CONCUR.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601 |

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)